



INF 1366 – Computação Gráfica Interativa

Modelagem Geométrica

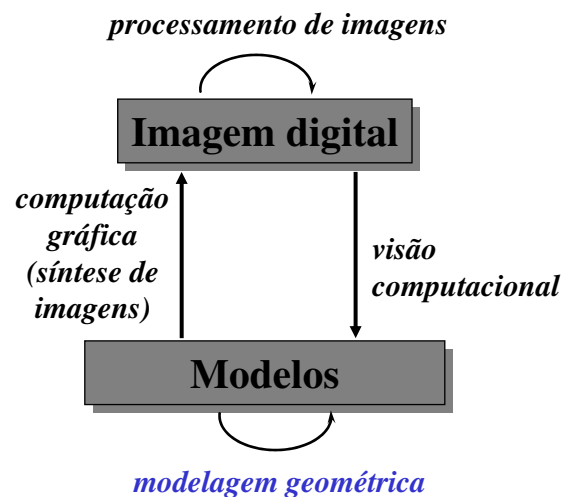
Alberto B. Raposo

abraposo@tecgraf.puc-rio.br

<http://www.tecgraf.puc-rio.br/~abraposo/INF1366/index.htm>

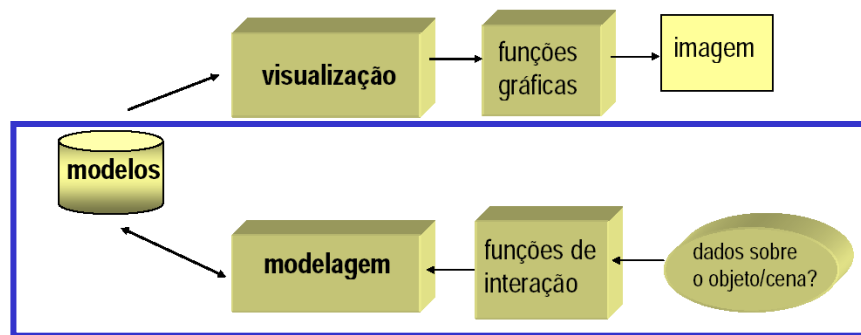
Alberto Raposo – PUC-Rio

Computação Gráfica e Áreas Correlatas



Alberto Raposo – PUC-Rio

Estrutura de aplicação gráfica interativa tradicional



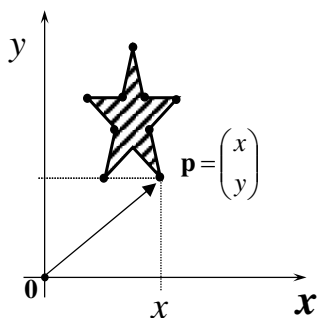
Aula de hoje (e as próximas)

Carla Freitas, UFRGS

Alberto Raposo – PUC-Rio

Espaços de Coordenadas

- Plano ou R^2 (2D)



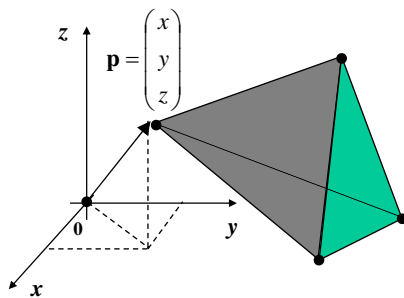
$$R^2 = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \text{ tal que } x, y \in R \right\}$$

Marcelo Gattass, PUC-Rio

Alberto Raposo – PUC-Rio

Espaços de Coordenadas

- Espaço ou R^3 (3D)



$$R^3 = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{ tal que } x, y, z \in R \right\}$$

Marcelo Gattass, PUC-Rio

Alberto Raposo – PUC-Rio

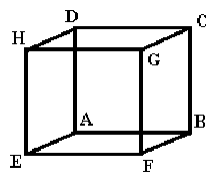
Modelagem Geométrica

- Tipos de estruturação de dados
 - Wireframe (representação de arestas)
 - Boundary representation (B-Rep)
 - Quadtree / Octree
- Malhas de Polígonos
 - LOD (nível de detalhe)
- Curvas
- Geração de 3D a partir de 2D
- Outras técnicas
 - Metaballs
 - Subdivision Surfaces
 - Low-Poly

Alberto Raposo – PUC-Rio

Wireframe

- Representação de arestas (pontos + conexões entre pontos)



Vértices

A (0, 0, 0)
 B (1, 0, 0)
 C (1, 1, 0)
 D (0, 1, 0)
 E (0, 0, 1)
 F (1, 0, 1)
 G (1, 1, 1)
 H (0, 1, 1)

Linhas

AB GH
 BC HE
 CD AE
 DA BF
 EF CG
 FG DH

Alberto Raposo – PUC-Rio

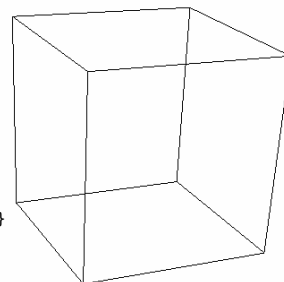
Wireframe em VRML: IndexedLineSet

#VRML V2.0 utf8

```
Transform { children [
  Shape {
    geometry IndexedLineSet {
      coord Coordinate {
        point [ 0 0 0, 1 0 0, 1 1 0, 0 1 0,
                0 0 1, 1 0 1, 1 1 1, 0 1 1 ] }
      coordIndex [ 0 1 -1 6 7 -1 1 2 -1 7 4 -1
                  2 3 -1 0 4 -1 3 0 -1 1 5 -1
                  4 5 -1 2 6 -1 5 6 -1 3 7 -1 ]

      color Color { color [ 0 0 0, 0 0 0, 0 0 0, 0 0 0,
                           0 0 0, 0 0 0, 0 0 0, 0 0 0,
                           0 0 0, 0 0 0, 0 0 0, 0 0 0 ] }
    }
  }
] } # end of children and Transform

Background {skyColor 1 1 1}
```

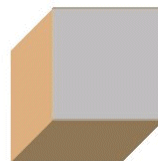
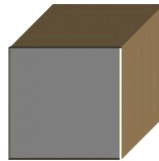
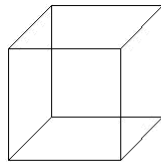


Alberto Raposo – PUC-Rio

Wireframe

- Vantagens
 - Simplicidade e velocidade na visualização dos modelos (geram-se apenas linhas)
- Problemas
 - Dificuldade de realizar operações com sólidos (cálculo de massa, volume, determinação de inclusão...)
 - Representação ambígua (sujeita a interpretações diferentes)

As duas representações abaixo são válidas para o modelo em wireframe à esquerda



Alberto Raposo – PUC-Rio

Márcio Pinho, PUCRS

Boundary Representation (B-Rep)

- Define-se o modelo 3D a partir de conjunto de polígonos que delimitam uma região fechada no espaço
 - Esses polígonos são as faces do objeto 3D (poliedro)



Alberto Raposo – PUC-Rio

Boundary Representation (B-Rep)

- Representações

- 1 lista de vértices explícita:

FACE: (x1, y1, z1)-(x2, y2, z2)- - (xn, yn, zn);

- 2 listas: lista de vértices e lista das topologias das faces (caso do IndexedFaceSet - VRML)

VÉRTICES:

1 - (x1, y1, z1)

2 - (x2, y2, z2)

....

n - (xn, yn, zn)

FACES:

1 - v1, v2, v3,, vn

2 - v3, v5,, vn

....

n - vn, v4,, v1

- 3 listas: vértices, arestas e faces

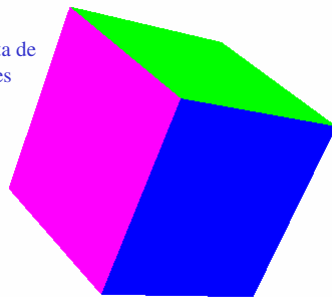
Alberto Raposo – PUC-Rio

B-Rep em VRML: IndexedFaceSet (2 listas)

```
#VRML V2.0 utf8
Transform { children [
  Shape {
    geometry IndexedFaceSet {
      coord Coordinate {
        point [ 0 0 0, 1 0 0, 1 1 0, 0 1 0,
                0 0 1, 1 0 1, 1 1 1, 0 1 1 ] }
      coordIndex [ 0 1 5 4 -1 3 7 6 2 -1
                  6 7 4 5 -1 7 3 0 4 -1
                  3 2 1 0 -1 2 6 5 1 -1 ]
      color Color { color [ 1 0 0, 0 1 0, 0 0 1,
                          1 0 1, 1 1 0, 0 1 1 ] }
      colorPerVertex FALSE
      colorIndex [ 0, 1, 2, 3, 4, 5 ]
    }
  }
] } # end of children and Transform
Background {skyColor 1 1 1}
```

Lista de vértices

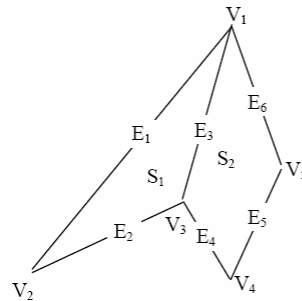
Lista de faces



Alberto Raposo – PUC-Rio

Exemplo de 3 listas

<http://gbdi.icmc.usp.br/documentacao/apostilas/cg/downloads/modpoliedrais.pdf>



Vertex Table

V₁: x₁,y₁,z₁
V₂: x₂,y₂,z₂
V₃: x₃,y₃,z₃
V₄: x₄,y₄,z₄
V₅: x₅,y₅,z₅

EdgeTable

E₁: V₁,V₂
E₂: V₂,V₃
E₃: V₃,V₁
E₄: V₃,V₄
E₅: V₄,V₅
E₆: V₅,V₁

Polygon-Surface Table

S₁: E₁,E₂,E₃
S₂: E₃,E₄,E₅,E₆

Alberto Raposo – PUC-Rio

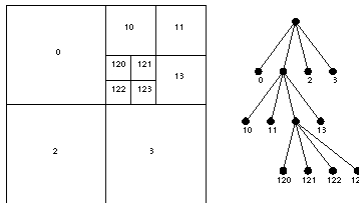
Quadtrees (2D) / Octrees (3D)

- Estruturas de dados (árvores) para decomposição hierárquica do plano (quadtrees) / espaço (octrees)
- Podem ser usadas para guardar diferentes tipos de dados, por ex.
 - Conjunto de pontos
 - Malhas poligonais

Alberto Raposo – PUC-Rio

Quadrees

- Todo nó representa um quadrado no plano.
- Todo nó interno possui exatamente quatro filhos, os quais representam os quatro quadrantes do nó pai: noroeste, nordeste, sudoeste e sudeste.
- A subdivisão continua conforme algum critério de parada.

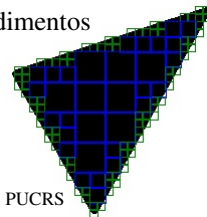


<http://www.tecgraf.puc-rio.br/~hermann/gc/>

Alberto Raposo – PUC-Rio

Quadtree – Critério de Parada

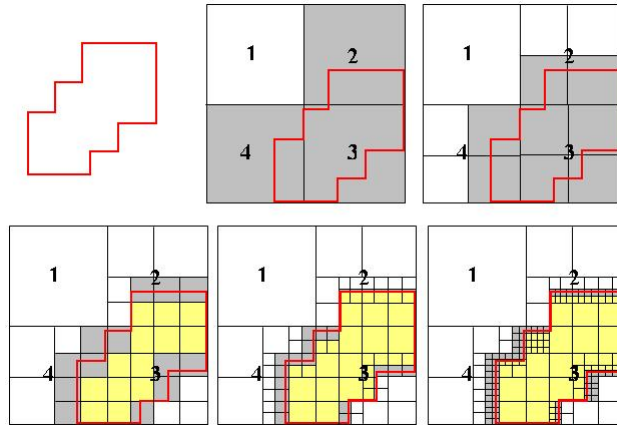
- Exemplo:
 1. Começa com quadrado envolvendo todo o objeto, que em seguida é dividido em 4 quadrados menores.
 2. Cada um é classificado em
 - Cheio:** o quadrado está totalmente dentro do objeto
 - Vazio:** o quadrado está totalmente fora do objeto
 - Cheio-Vazio:** apenas parte do quadrado é ocupada pelo objeto
 3. Para cada quadrado cheio-vazio, repetir os procedimentos 1 e 2.
- O procedimento encerra quando só existirem quadrados cheios e vazios



Alberto Raposo – PUC-Rio

Pinho, PUCRS

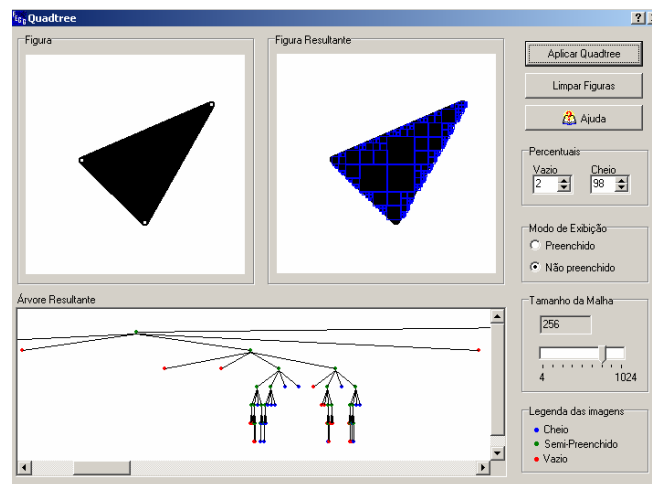
Quadtree - Exemplo



<http://lcp.lcad.icmc.usp.br/~nonato/ED/Quadtree/quadtree.htm>

Alberto Raposo – PUC-Rio

Quadtree – Programa Exemplo

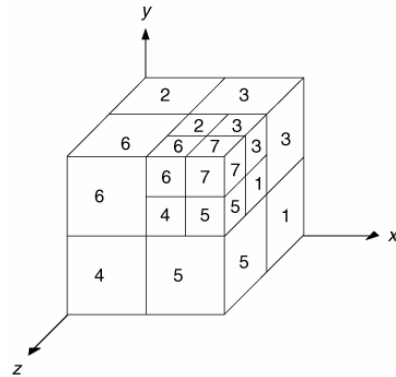


Alberto Raposo – PUC-Rio

Autores: Patrícia Zottis e Rodrigo Fehse
Alterações: Leonardo Langie - PUCRS

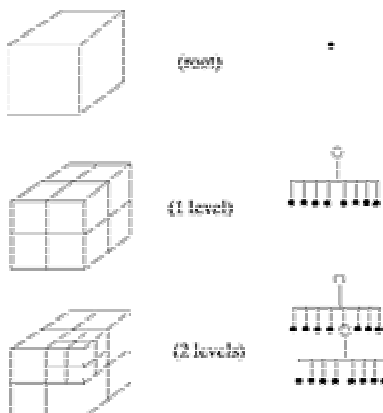
Octree

- Idêntica à Quadtree, mas considerando o espaço 3D
 - Cubo é dividido em 8 sub-cubos



Alberto Raposo – PUC-Rio

Octrees



Alberto Raposo – PUC-Rio

Octree

- Algoritmo de construção em C

Pinho, PUCRS

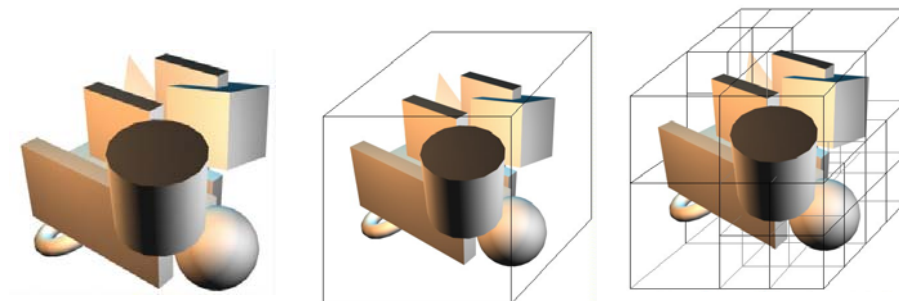
Alberto Raposo – PUC-Rio

```
typedef struct octree
{
    XYZR Min, Max; /* Limites do nodo */
    char codigo; /* PRETO, BRANCO, CINZA */
    struct octree *oct[8]; /* 8 nodos-filhos */
}OCTREE;

/* PRETO : nodo TODO ocupado pelo objeto */
/* BRANCO : nodo NÃO ocupado pelo objeto */
/* CINZA : nodo PARCIALMENTE ocupado pelo objeto */

CriaArvore(Sólido, OCTREE *raiz, nivel)
(
    C = Classifica(Objeto, raiz->Min, raiz->Max);
    if (C = PRETO) raiz->codigo = PRETO;
    if (C = BRANCO) raiz->codigo = BRANCO;
    if (C = CINZA)
    {
        raiz->codigo = CINZA;
        subdivide(raiz); /* gera os oito filhos */
        for(i=0;i<8;i++)
        { /* chama a função de criação */
            CriaArvore(Sólido, raiz->oct[i], nivel-1);
        }
    }
}
```

Octrees

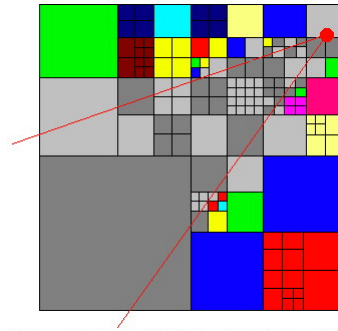


FlipCode.com

Alberto Raposo – PUC-Rio

Uso de Octrees e Quadrees

- Exemplos:
 - Frustum culling, detecção de colisão, operações de união e interseção
 - Se pai (não) é importante, todos os filhos também (não) são
- Desvantagem:
 - Trabalhosas para manipular

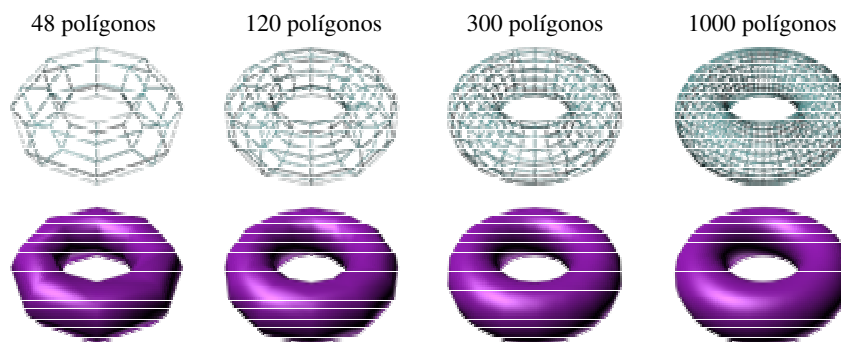


FlipCode.com

Alberto Raposo – PUC-Rio

Malhas de Polígonos

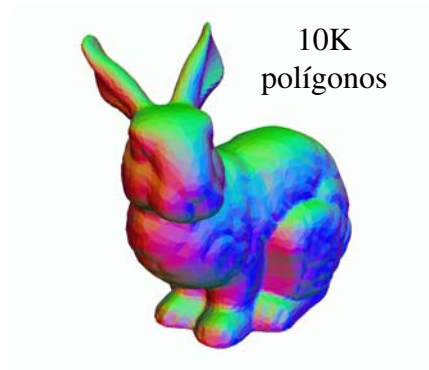
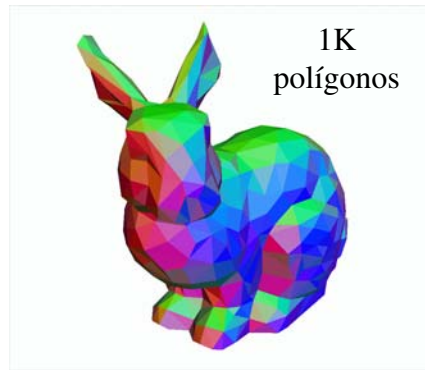
- Construção de modelos 3D usando grupos de polígonos.
 - Como cada polígono é planar, necessita-se grande quantidade de polígonos para dar a impressão de superfícies curvas



Alberto Raposo – PUC-Rio

John Dingliana, 2004

Malhas de Polígonos



MIT EECS 6.837, Durand and Cutler

Alberto Raposo – PUC-Rio

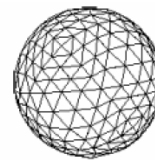
Mesh Tessellation

- Construção de malhas poligonais

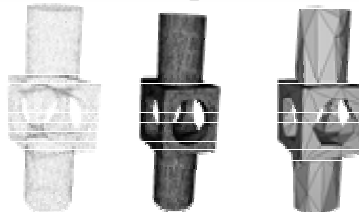
- A partir de representações abstratas

<http://www.cs.lth.se/Education/Courses/EDA221/>

center: (5,2,3) →
radius: 14



- A partir de núvens de pontos

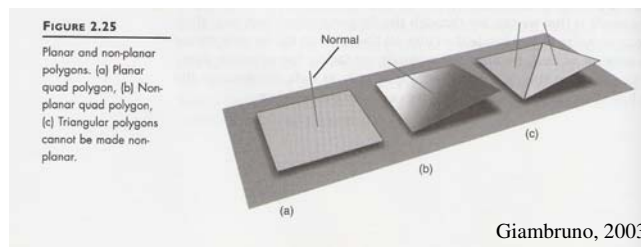


<http://www.ticam.utexas.edu/CCV/projects/VisualEyes/visualization/geomod/cloud/cloud.html>

Alberto Raposo – PUC-Rio

Malhas de triângulos

- Costuma-se usar triângulos como o polígono das malhas
 - O polígono é gerado com exatamente 3 vértices por face
 - Um vértice pode pertencer a qualquer número de faces
 - Adjacência calculada em tempo constante
 - Triângulos são sempre planares

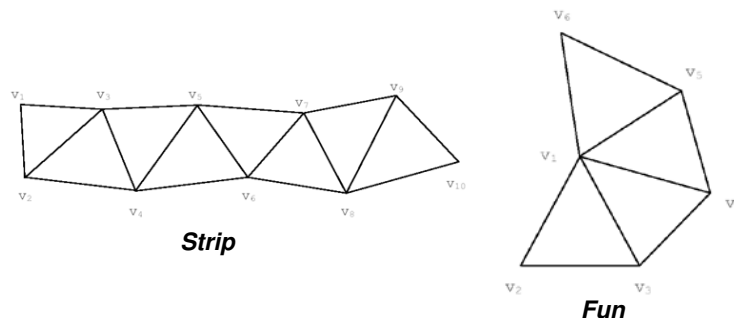


Giamb Bruno, 2003

Alberto Raposo – PUC-Rio

Alguns tipos de malhas de triângulos

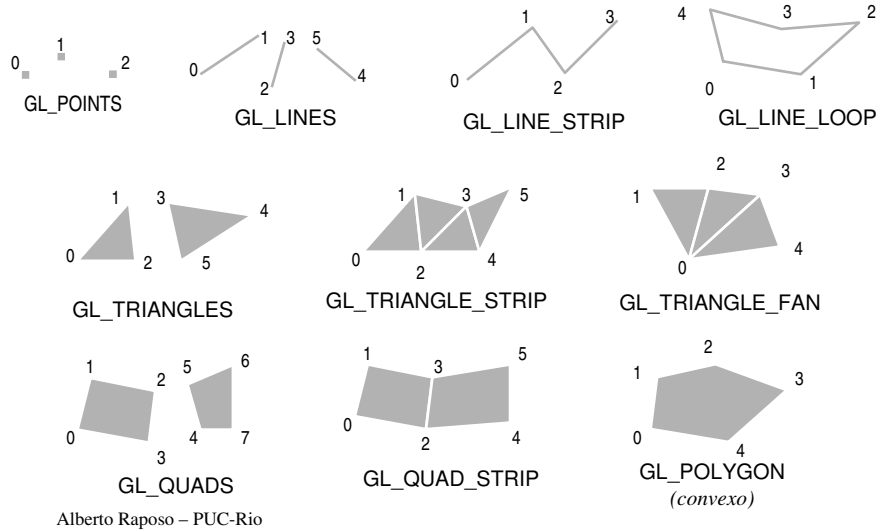
- O primeiro triângulo é desenhado com três vértices, e os demais com apenas um.



- Muitos dos vértices são comuns a vários polígonos que o constituem. Assim, a organização dos polígonos com vista o compartilhamento dos vértices comuns traduz-se num envio e processamento únicos destes vértices [Möller 2003].

Alberto Raposo – PUC-Rio

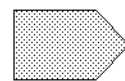
Tipos de primitivas em OpenGL



Exemplo em OpenGL

```
glBegin(tipo_de_prim);
    ...define atributo de vértice
    ...define vértice
glEnd();
```

```
glBegin(GL_POLYGON);
    glVertex2f(0.0, 0.0);
    glVertex2f(0.0, 3.0);
    glVertex2f(3.0, 3.0);
    glVertex2f(4.0, 1.5);
    glVertex2f(3.0, 0.0);
glEnd();
```



GL_POLYGON



GL_POINTS

Alberto Raposo – PUC-Rio

Problema Geral

- Quantos mais polígonos, menos facetada fica a superfície curva
 - Mais polígonos, significa mais tempo de processamento!!!



(menos polígonos)

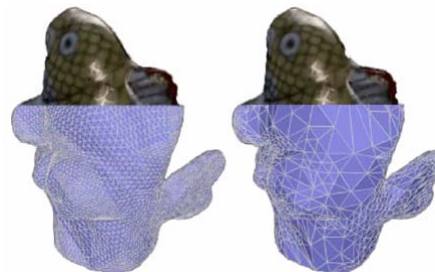


(mais polígonos)

Alberto Raposo – PUC-Rio

LOD – Level of Detail

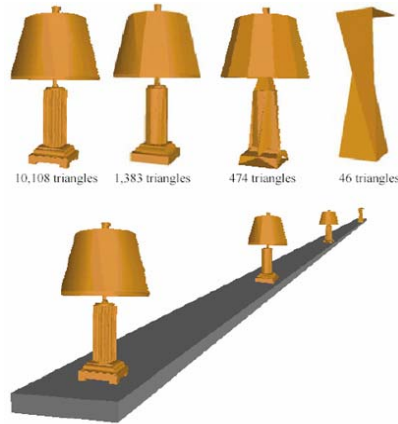
- À medida que a distância da câmera a um modelo aumenta, o espaço por este ocupado na janela diminui e, conseqüentemente, o detalhe com que é visualizado também diminui.
- O LOD permite definir representações alternativas para um objeto gráfico, cada uma sendo ativada de acordo com a distância ao observador.



Alberto Raposo – PUC-Rio

LOD

- Torna-se desnecessário e ineficiente definir o objeto com todo detalhe. O objetivo principal é o de utilizar diferentes representações de um modelo, normalmente de resoluções distintas, que serão selecionadas de acordo com um critério de decisão pré-determinado. Um dos critérios de decisão mais utilizado é a distância do modelo à câmera.



Alberto Raposo – PUC-Rio

Tipos de LOD

- Discreto
 - A construção das diferentes representações do modelo é realizada numa fase de pré-processamento, sendo associada a cada uma delas um intervalo de distâncias à câmera dentro do qual o nível de detalhe deve ser utilizado.
 - Durante a execução, o algoritmo calcula a distância da câmera ao objeto e avalia qual dos diferentes níveis de detalhe deve ser utilizado.

Alberto Raposo – PUC-Rio

Tipos de LOD

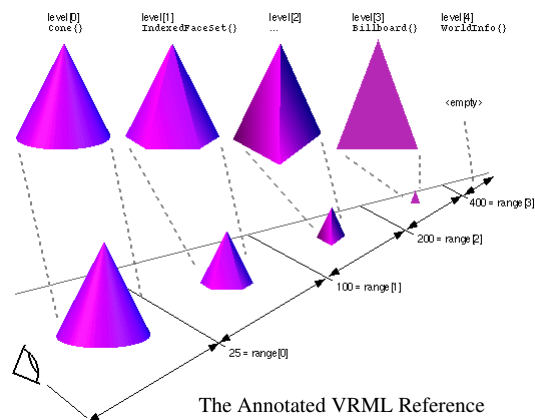
- Contínuo
 - Os níveis de detalhe são gerados em tempo de execução.
 - View dependent LOD
 - Extensão de LOD contínuo usando posição do observador para definir o nível de detalhe.

Alberto Raposo – PUC-Rio

Exemplo de LOD Discreto

- VRML: nó LOD

```
LOD {
  exposedField MFNode level []
  field SFVec3f center 0 0 0
  field MFFloat range []
}
```



Alberto Raposo – PUC-Rio

Exemplo LOD (VRML)

```
#VRML V2.0 utf8
LOD {
  range [ 25, 100, 200, 400 ]
  level [
    # level 0 - default gray, lit cone
    Transform { translation 0 1.5 0 children
      Shape {
        appearance DEF AP Appearance { material Material {} }
        geometry Cone { bottomRadius 1 height 3 }
      }
    }
    # level 1 - lit, 8 triangle cone approximation
    Shape {
      appearance USE AP
      geometry IndexedFaceSet {
        coord Coordinate {
          point [ 1 0 0, .707 0 -.707, 0 0 -1,
                  -.707 0 -.707, -1 0 0, -.707 0 .707, 0 0 1,
                  .707 0 .707, 0 3 0 ]
        }
        coordIndex [ 0 1 8 -1 1 2 8 -1 2 3 8 -1 3 4 8 -1
                     4 5 8 -1 5 6 8 -1 6 7 8 -1 7 0 8 -1
                     0 7 6 5 4 3 2 1 ]
      }
    }
  ]
}
```

Distâncias de cada nível

Nível 0: cone

Nível 1: cone facetado

Alberto Raposo – PUC-Rio

Exemplo LOD (VRML)

```
# level 2 - lit, tetrahedron
Shape {
  appearance USE AP
  geometry IndexedFaceSet {
    coord Coordinate {
      point [ 1 0 0, 0 0 -1, -1 0 0, 0 0 1, 0 3 0 ]
    }
    coordIndex [ 0 1 4 -1 1 2 4 -1 2 3 4 -1
                 3 0 4 -1 0 3 2 1 ]
  }
}
# level 3 - unlit, medium gray billboarded polygon
Billboard {
  children Shape {
    geometry IndexedFaceSet {
      coord Coordinate { point [ 1 0 0, 0 3 0, -1 0 0 ] }
      coordIndex [ 0 1 2 ]
      colorPerVertex FALSE
      color Color { color 0.5 0.5 0.5 }
    }
  }
}
# level 4 - empty
WorldInfo {}
}
```

Nível 2: cone com menos faces

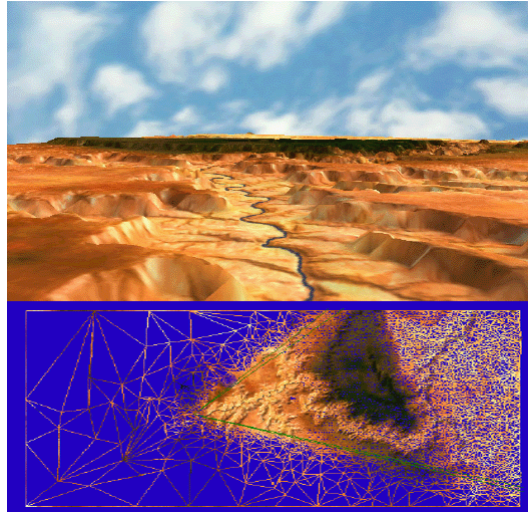
Nível 3: triângulo

Nível 4: nada

Alberto Raposo – PUC-Rio

LOD Contínuo

View dependent LOD

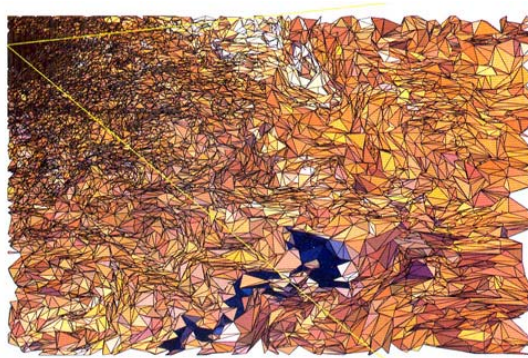


Alberto Raposo – PUC-Rio

LOD Contínuo

View dependent LOD

observador

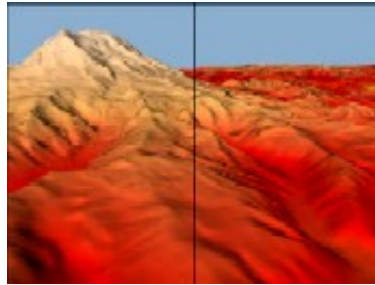


Alberto Raposo – PUC-Rio

LOD Contínuo

Visualização de Terrenos

- <http://www.llnl.gov/icc/sdd/img/images.shtml>

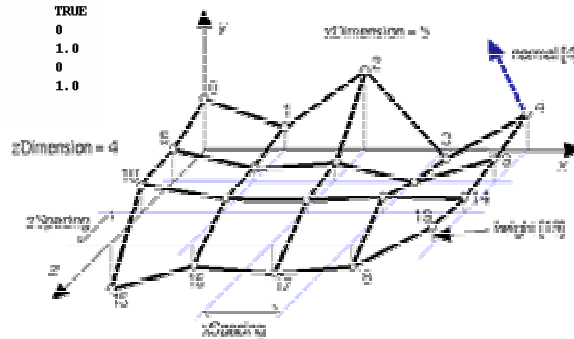


(vídeo)

Alberto Raposo – PUC-Rio

VRML: Elevation Grid

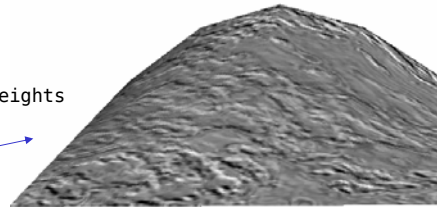
```
ElevationGrid {
  eventIn      MFFloat  set_height
  exposedField SFNode  color      NULL
  exposedField SFNode  normal     NULL
  exposedField SFNode  texCoord   NULL
  field        MFFloat  height     []
  field        SFBool   ccw        TRUE
  field        SFBool   colorPerVertex TRUE
  field        SFFloat  creaseAngle 0
  field        SFBool   normalPerVertex TRUE
  field        SFBool   solid       TRUE
  field        SFInt32  xDimension  0
  field        SFFloat  xSpacing    1.0
  field        SFInt32  zDimension  0
  field        SFFloat  zSpacing    1.0
}
```



Alberto Raposo – PUC-Rio

Exemplo de Elevation Grid

```
#VRML V2.0 utf8
Transform { children [
  Shape {
    geometry DEF EG ElevationGrid {
      xDimension 5
      xSpacing 1
      zDimension 4
      zSpacing 1
      height [           # 5x4 array of heights
        0 .707 1 .707 0
        0 .47 .667 .47 0
        0 .236 .33 .236 0
        0 0 0 0 0
      ]
      creaseAngle 0.8
    }
    appearance Appearance {
      material DEF M Material { diffuseColor 1 1 1 }
      texture DEF IT ImageTexture { url "stone.jpg" }
    }
  }
]
```



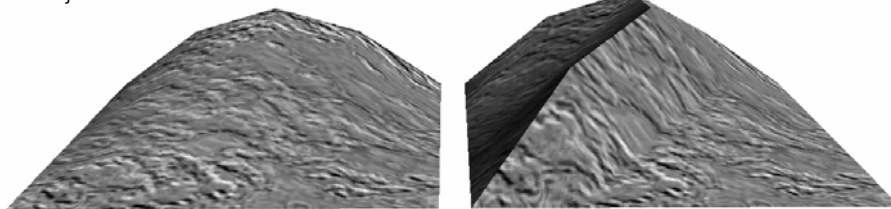
Alberto Raposo – PUC-Rio

Exemplo de Elevation Grid

```
Transform {
  translation 4.3 0 0
  children Shape {
    geometry ElevationGrid {
      xDimension 5
      xSpacing 1
      zDimension 4
      zSpacing 1
      height [ # 5x4 array of heights
        0 .876 1.2 .66 0
        0 .555 1.3 .47 0
        0 1. .33 .2 0
        0 0 0 0 0
      ]
      creaseAngle 0.8
    }
  }
}

appearance Appearance {
  material USE M
  texture USE IT
}

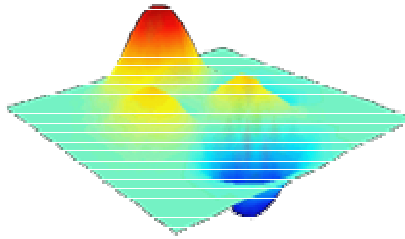
DirectionalLight { direction -0.80 -0.6 0 }
Viewpoint { position 3 2 8 }
Background { skyColor 1 1 1 }
```



Alberto Raposo – PUC-Rio

Elevation Grid

- Exemplo de superfície matemática

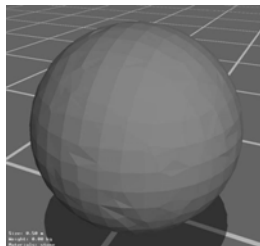


<http://pcf1.chembio.ntnu.no/~bka/div/vrml/elevation.html>

Alberto Raposo – PUC-Rio

Apesar de tudo...

- Superfícies poligonais são limitadas
 - Facetas planares
 - Deformação é difícil
 - Parametrização não é natural



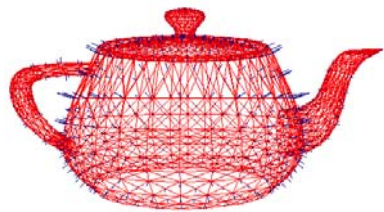
Alberto Raposo – PUC-Rio



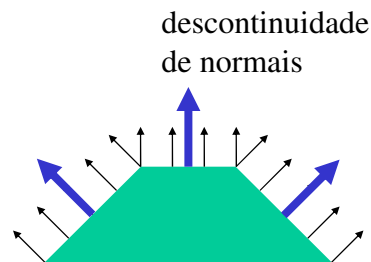
MIT EECS 6.837, Durand and Cutler

Porque o facetamento

- Shading (Gouraud) é feito a partir das normais de cada uma das superfícies (polígonos)



Alberto Raposo – PUC-Rio



MIT EECS 6.837, Durand and Cutler

Continuidade de curvas (2D) / superfícies (3D)

G^0 continuidade geométrica: 2 segmentos / superfícies conectadas

Não há buracos na curva / superfície

$G^0 \rightarrow C^0$ (continuidade paramétrica)

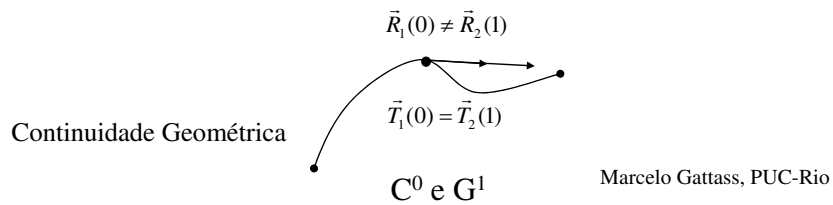
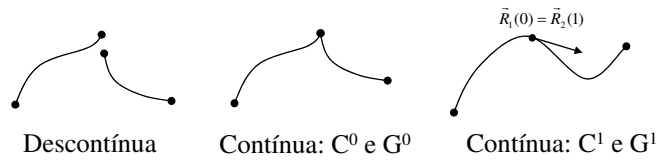
G^1 continuidade geométrica : a direção das tangentes dos 2 segmentos / superfícies são iguais no ponto / curva de junção

C^1 continuidade (paramétrica): vetores tangentes dos dois segmentos / superfícies são iguais em magnitude e direção no ponto / curva de junção

C^n continuidade (paramétrica): direção e magnitude da n-ésima derivada são iguais no ponto / curva de junção

Alberto Raposo – PUC-Rio

Continuidade de curvas (2D) / superfícies (3D)



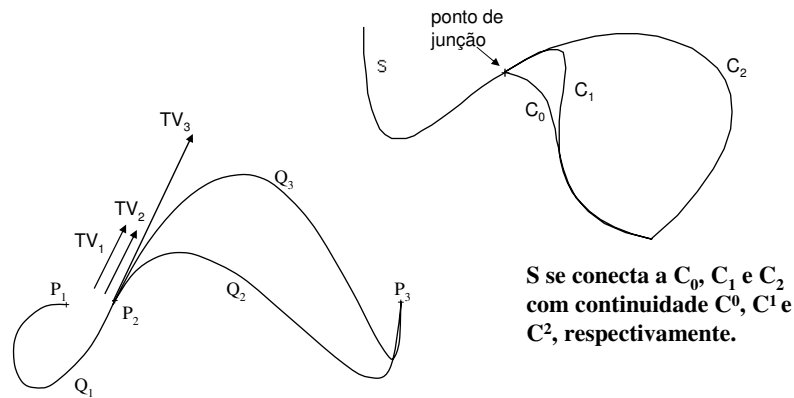
Alberto Raposo – PUC-Rio

Continuidade de curvas (2D) / superfícies (3D)

- Malhas de polígonos são C^0 (G^0) apenas
- Superfícies C^1 garante superfícies menos facetadas (smoothness)
- Superfícies C^2 são ainda mais “polidas” que as C^1

Alberto Raposo – PUC-Rio

Exemplos de Conexões de Curvas



S se conecta a C_0 , C_1 e C_2 com continuidade C^0 , C^1 e C^2 , respectivamente.

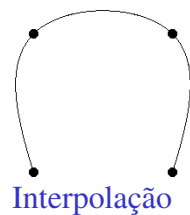
Q_1 e Q_2 têm continuidade C^1 porque seus vetores tangentes, TV_1 e TV_2 são iguais. Q_1 e Q_3 têm continuidade G^1 apenas.

Kessler, Dinh, 2003

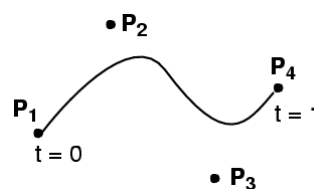
Alberto Raposo – PUC-Rio

Controle de Curvas / Splines

- Curvas são definidas por pontos de controle
- Alterando os pontos, altera-se a curva

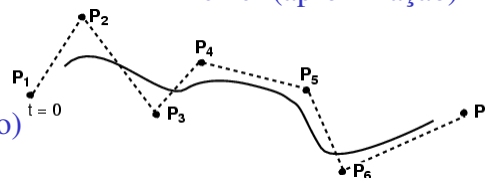


Interpolação



Bézier (aproximação)

BSpline
(aproximação)

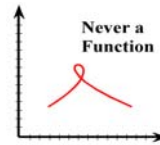


Alberto Raposo – PUC-Rio

MIT EECS 6.837, Durand and Cutler

Funções

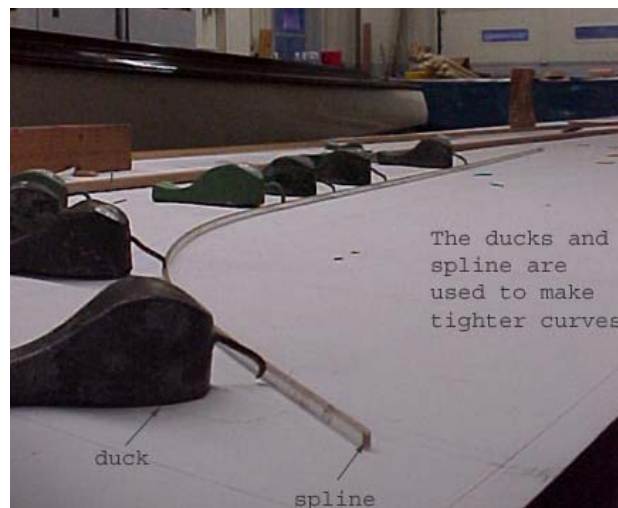
- Explícitas: $y = f(x)$ [e.g. $y=2x^2$]
 - Apenas 1 valor de y para cada x
- Implícitas: $f(x,y)=0$ [e.g. $x^2+y^2-r^2=0$]
 - Precisa de restrições para modelar apenas partes da curva
 - Manter continuidade na junção de 2 curvas é difícil
- Paramétricas: $x=f(t)$, $y=f(t)$ [e.g. $x=t^3+3$, $y=3t^2+2t+1$]
 - Curvaturas representadas como vetores tangentes (d/dt).
 - Fácil manter continuidade nas junções



Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

Curvas



<http://www.abm.org>

Alberto Raposo – PUC-Rio

Curvas Paramétricas

Para selecionar parte da curva: $0 \leq t \leq 1$

Linear:	Quadrática:	Cúbica:
$x = a_x t + b_x$	$x = a_x t^2 + b_x t + c_x$	$x = a_x t^3 + b_x t^2 + c_x t + d_x$
$y = a_y t + b_y$	$y = a_y t^2 + b_y t + c_y$	$y = a_y t^3 + b_y t^2 + c_y t + d_y$
$z = a_z t + b_z$	$z = a_z t^2 + b_z t + c_z$	$z = a_z t^3 + b_z t^2 + c_z t + d_z$

Em CG, preferem-se as cúbicas, que provêem um balanceamento entre *flexibilidade* e *complexidade* na especificação e computação da forma.

- Precisa 4 pontos/derivadas conhecidas para determinar 4 coeficientes desconhecidos.

Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

Equações Paramétricas

$$x = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$z = a_z t^3 + b_z t^2 + c_z t + d_z$$

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = TC$$

$$T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \quad C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

Matriz Base e Matriz Geométrica

$$Q(t) = G M T$$

$$\begin{bmatrix} G_1 & G_2 & G_3 & G_4 \end{bmatrix} \begin{bmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & m_{44} \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Matriz Geométrica Matriz Base Matriz T

- Idéia:
 - Curvas diferentes podem ser especificadas alterando-se a informação geométrica na matriz G.
 - A matriz base tem valores constantes específicos de cada família de curvas.

Alberto Raposo – PUC-Rio

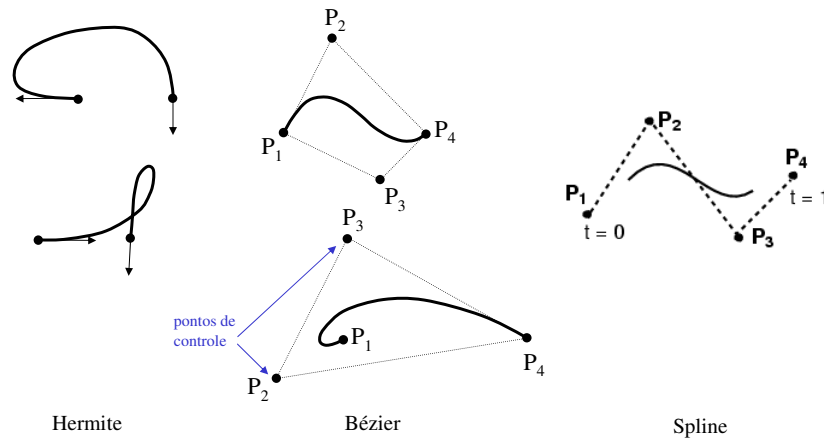
Famílias de Curvas

Família	Tipo	Definida por
Hermite	Cúbica	2 pontos extremos, vetores tangentes nos extremos
Bézier	Cúbica	2 pontos extremos, 2 pontos de controle
Splines	Cúbica	4 pontos de controle

Alberto Raposo – PUC-Rio

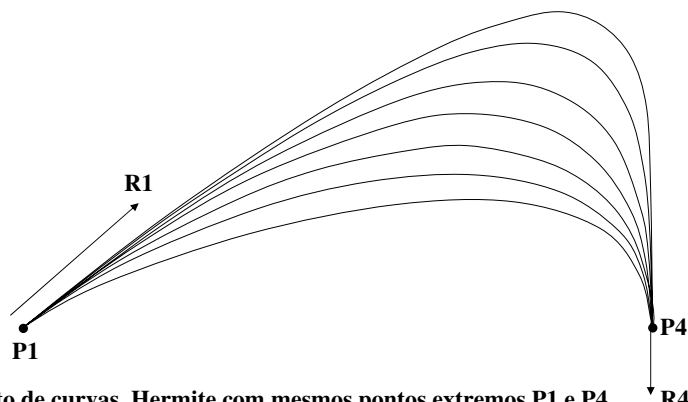
Kessler, Dinh, 2003

Famílias de Curvas



Alberto Raposo – PUC-Rio

Exemplos Hermite

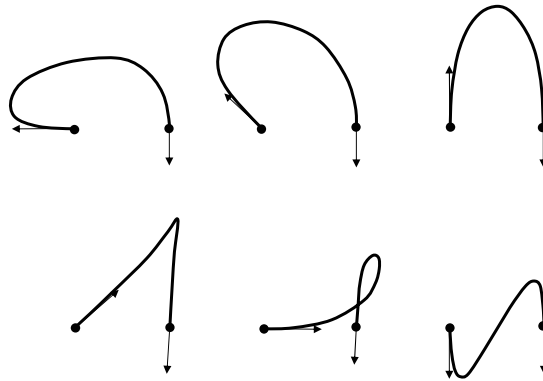


Conjunto de curvas Hermite com mesmos pontos extremos P_1 e P_4 , vetores tangente R_1 e R_4 com mesma direção, mas magnitudes diferentes de R_1 . A magnitude de R_4 é mantida fixa.

Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

Exemplos Hermite



Curvas com pontos extremos fixos e magnitudes dos vetores tangentes iguais, mas a direção do vetor tangente da esquerda varia.

Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

Formulação Hermite

$$Q(t) = G M T$$

$$\begin{bmatrix} G_1 & G_2 & G_3 & G_4 \end{bmatrix}$$

Matriz Geométrica

$$\begin{bmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & m_{44} \end{bmatrix}$$

Matriz Base

$$\begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Matriz T

Alberto Raposo – PUC-Rio

Formulação Hermite

$$Q(t) = G_H M_H T = [P_1 \ P_4 \ R_1 \ R_4] M_H \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$P_1 = Q(0) = G_H M_H \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$R_1 = Q'(0) = G_H M_H \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$P_4 = Q(1) = G_H M_H \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$R_4 = Q'(1) = G_H M_H \begin{bmatrix} 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

Formulação Hermite

$$[P_1 \ P_4 \ R_1 \ R_4] = [P_1 \ P_4 \ R_1 \ R_4] M_H \begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$M_H = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

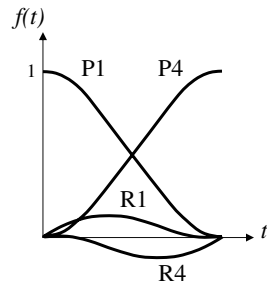
Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

Formulação Hermite

$$Q(t) = G_H M_H T$$

$Q(t)$ é soma ponderada dos elementos de G_H



Função blending de Hermite

Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$

Formulação Hermite

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$



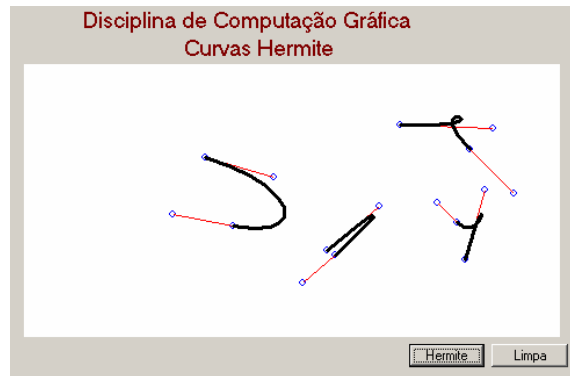
$$x(t) = (2t^3 - 3t^2 + 1)P_1 x + (-2t^3 + 3t^2)P_4 x + (t^3 - 2t^2 + t)R_1 x + (t^3 - t^2)R_4 x$$

$$y(t) = (2t^3 - 3t^2 + 1)P_1 y + (-2t^3 + 3t^2)P_4 y + (t^3 - 2t^2 + t)R_1 y + (t^3 - t^2)R_4 y$$

$$z(t) = (2t^3 - 3t^2 + 1)P_1 z + (-2t^3 + 3t^2)P_4 z + (t^3 - 2t^2 + t)R_1 z + (t^3 - t^2)R_4 z$$

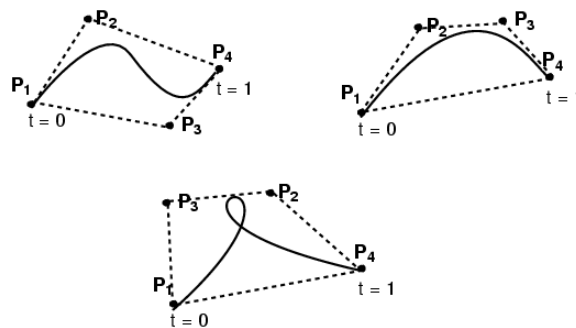
Alberto Raposo – PUC-Rio

Programa Hermite



Alberto Raposo – PUC-Rio

Exemplos Bézier



Marcelo Walter, Unisinos

- 4 pontos
 - Curva passa pelo primeiro e pelo último
 - P_2 e P_3 definem as tangentes em P_1 e P_4 :
 - $R_1 = 3(P_2 - P_1)$ e $R_4 = 3(P_4 - P_3)$

Alberto Raposo – PUC-Rio

■ A razão para usar a constante 3:

- Considere a curva designada pelos pontos: $(0,0)$, $(0,1)$, $(0,2)$, $(0,3)$.



- A definição desta curva é: $p(t) = P_1 + t(P_4 - P_1)$

- Logo: $p'(t) = P_4 - P_1$

- Se a velocidade for constante:

$$R_1 = p'(0) = P_4 - P_1 = 3(P_2 - P_1) \quad R_4 = p'(1) = P_4 - P_1 = 3(P_4 - P_3)$$

Formulação Bézier

$$Q(t) = G_B M_B T \quad G_B = [P_1 \ P_2 \ P_3 \ P_4] \quad T = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Lembrando que: $R_1 = 3(P_2 - P_1)$ e $R_4 = 3(P_4 - P_3)$
podemos relacionar Bézier com Hermite:

$$G_H = [P_1 \ P_4 \ R_1 \ R_4] = [P_1 \ P_2 \ P_3 \ P_4] \begin{bmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{bmatrix} = G_B M_{HB}$$

Alberto Raposo – PUC-Rio

Formulação Bézier

Voltando a Hermite: $Q(t) = G_H M_H T$

Como: $G_H = G_B M_{HB}$

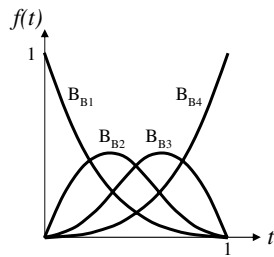
Temos: $Q(t) = G_B M_{HB} M_H T$

$$M_B = M_H \cdot M_{HB} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Alberto Raposo – PUC-Rio

Formulação Bézier

$$Q(t) = G_B M_B T \quad Q(t) = (-t^3 + 3t^2 - 3t + 1)P_1 + (3t^3 - 6t^2 + 3t)P_2 + (-3t^3 + 3t^2)P_3 + t^3 P_4$$



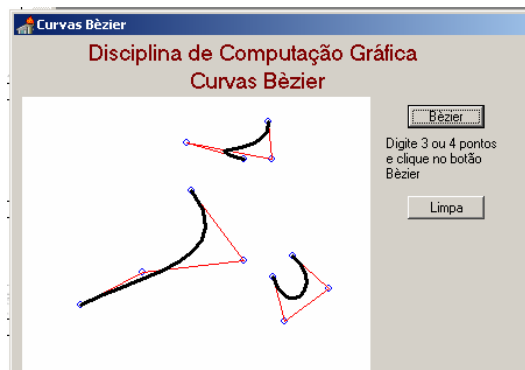
Polinômios de Bernstein:

$$B_{B_1} = (1-t)^3 \quad B_{B_2} = 3t(1-t)^2$$

$$B_{B_3} = 3t^2(1-t) \quad B_{B_4} = t^3$$

Alberto Raposo – PUC-Rio

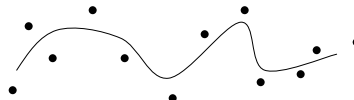
Programa Bézier



Alberto Raposo – PUC-Rio

Splines

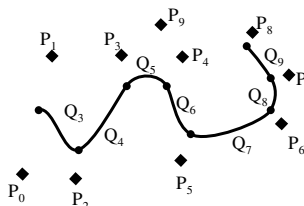
- Junções em curvas Hermite e Bézier são facilmente C^1 e G^1 , mas garantir C^2 não é trivial.
- Spline é curva que garante C^2
 - C^2 é útil quando curva trilha caminho da câmera (pense como velocidade: C^1 e aceleração: C^2)
- Pode passar ou não pelos pontos de controle



Alberto Raposo – PUC-Rio

Natural vs. B-Spline

- Splines naturais
 - n pontos de controle, que afetam toda a curva
 - difícil computação
- B-Splines
 - curva com $m+1$ pontos de controle, $P_0, P_1, \dots, P_m$, $m \geq 3$, definindo $m-2$ segmentos polinomiais cúbicos conectados
 - segmento Q_i é definido por $P_{i-3}, P_{i-2}, P_{i-1}$ e P_i
 - efeito dos pontos de controle é localizado (restrito a 4 segmentos)



Alberto Raposo – PUC-Rio

Kessler, Dinh, 2003

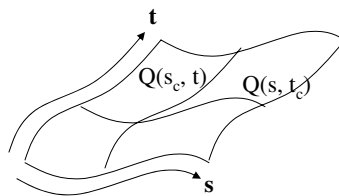
B-Splines

- *Uniform B-Splines* têm nós em intervalos iguais de t .
 - Distâncias em t entre nós adjacentes é a mesma.
 - Funções blending são as mesmas para todos os segmentos.
- *Nonuniform B-Splines* têm intervalos diferentes de t entre os nós.
- *Nonuniform Rational B-Splines* (NURBS) são comumente usadas em modelagem 3D.
 - Curvas são invariantes às transformações perspectivas.

Alberto Raposo – PUC-Rio

Extensão para Superfícies

- Toda a matemática das curvas paramétricas cúbicas pode ser estendida para superfícies.
 - *Superfícies Paramétricas bicúbicas*
- Usa-se 2 parâmetros, s e t , ao invés de apenas t : $Q(s, t)$



- Superfície definida por 16 coeficientes (e 16 valores conhecidos).

Alberto Raposo – PUC-Rio

Splines VRML (Extensão Cortona)

- SplineCone
- SplineCylinder
- SplineElevationGrid
- SplineExtrusion
- SplineFaceSet
- SplineSphere

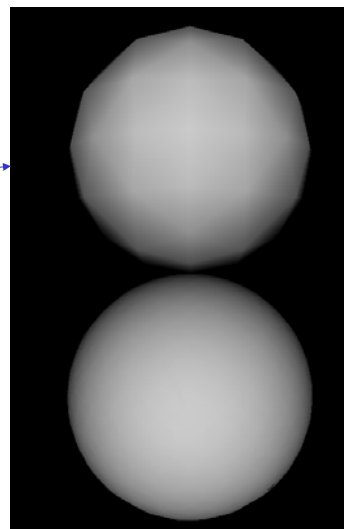
```
SplineSphere {  
  field SFFloat radius 1  
  exposedField MFFloat distance 10  
  exposedField MFFloat quality [0, 0.75]  
}
```

“LOD” embutido: a partir da distância especificada, a curva passa a ter a percentagem de qualidade

Alberto Raposo – PUC-Rio

Splines VRML (Extensão Cortona)

```
#VRML V2.0 utf8  
Viewpoint { description "Initial view" position 0 0 9 }  
NavigationInfo { type "EXAMINE" }  
  
# No Spline  
Transform {  
  children Shape {  
    appearance Appearance { material Material { } }  
    geometry Sphere {  
      radius 2  
    }  
  }  
}  
  
# Spline - Cortona Extension  
Transform {  
  translation 0 -4 0  
  children Shape {  
    appearance Appearance { material Material { } }  
    geometry SplineSphere {  
      radius 2  
      distance 5  
      quality [1 0.5]  
    }  
  }  
}
```



Alberto Raposo – PUC-Rio

Splines: Demo em VRML



<http://www.parallelgraphics.com>

Alberto Raposo – PUC-Rio

NURBS: VRML

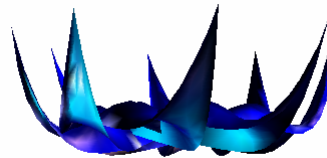
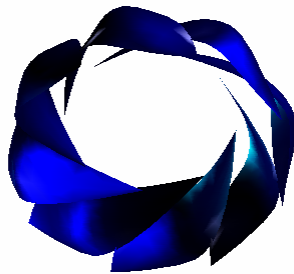
```

NurbsSurface {
  field      SFInt32  uDimension 0    #[-inf, inf)
  field      SFInt32  vDimension 0    #[-inf, inf)
  field      MFFloat  uKnot  []       #[-inf, inf)
  field      MFFloat  vKnot  []       #[-inf, inf)
  field      SFInt32  uOrder  3        #[-inf, inf)
  field      SFInt32  vOrder  3        #[-inf, inf)
  exposedField MFVec3f controlPoint [] #[-inf, inf)
  exposedField MFFloat weight  []      #[-inf, inf)
  exposedField SFInt32 uTessellation 0 #[-inf, inf)
  exposedField SFInt32 vTessellation 0 #[-inf, inf)
  exposedField SFNode texCoord  []
  exposedField SFBool  ccw  TRUE
  exposedField SFBool  solid TRUE
  exposedField MFFloat distance 10
  exposedField MFFloat quality [0, 0.75]
}
    
```

} número de pontos de controle em cada dimensão
 } vetores de nós
 } grau dos polinômios = ordem -1 (ex. para spline cúbica, ordem = 4)
 } pontos de controle (uDimension x vDimension)
 } peso de cada ponto de controle

Alberto Raposo – PUC-Rio

NURBS: Demo em VRML

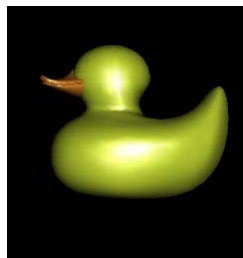


<http://www.parallelgraphics.com>

Alberto Raposo – PUC-Rio

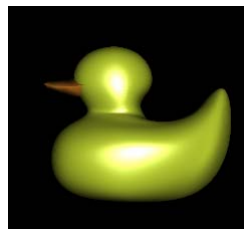
Comparação

- Armazenado como NURB (11KB)



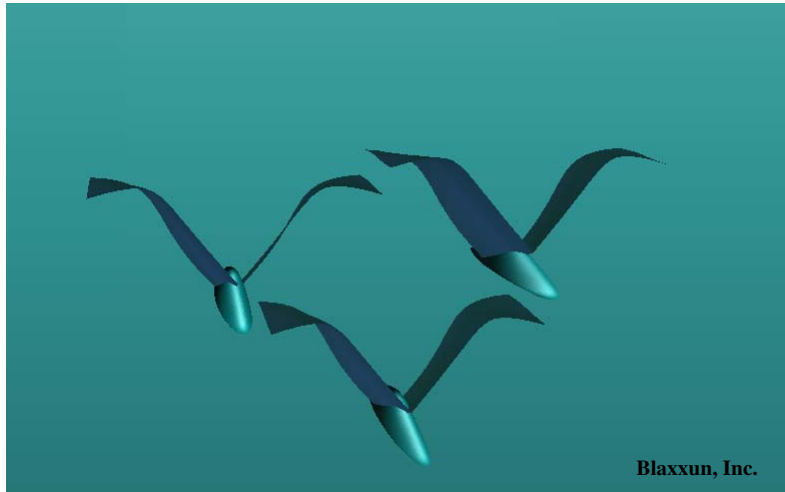
Alberto Raposo – PUC-Rio

- Armazenado como IndexFaceSet de alta resolução (2.2MB)



Blaxxun, Inc.

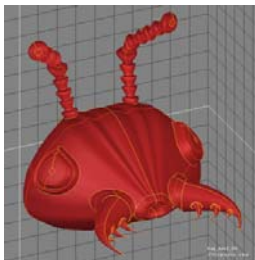
Demo



Blaxxun, Inc.

Alberto Raposo – PUC-Rio

Superfícies NURBs



<http://www.tiemdesign.com>



Stephane, Paris



Alberto Raposo – PUC-Rio

Geração de 3D a partir de 2D

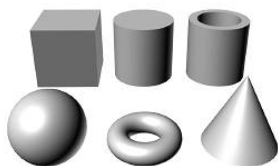
- Primitivas 3D
- CSG (Constructive Solid Geometry)
- Extrusão
- Lathing (revolução)
- Sweeping (extrusão ao longo de uma curva)
- Skinning (sweeping com cortes variados)

Alberto Raposo – PUC-Rio

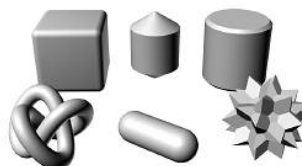
Primitivas 3D

- Formas geométricas 3D básicas, que podem ser estendidas por operações booleanas (CSG)

Primitivas básicas



Primitivas "menos básicas"

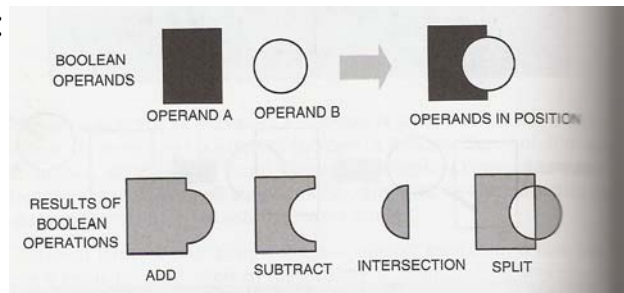


Alberto Raposo – PUC-Rio

Giambruno, 2003

CSG (Constructive Solid Geometry)

- Sólidos montados a partir de operações booleanas com outros sólidos
- No plano:

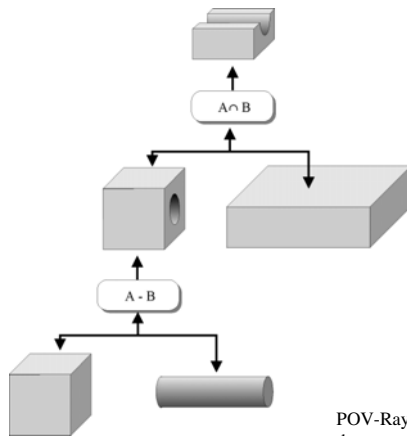


Alberto Raposo – PUC-Rio

Giambruno, 2003

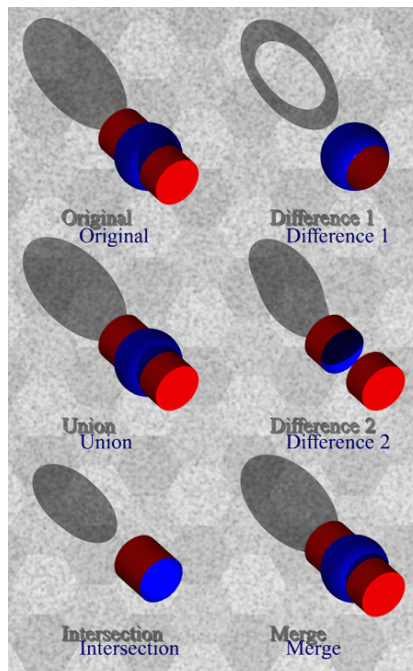
CSG

- No espaço:

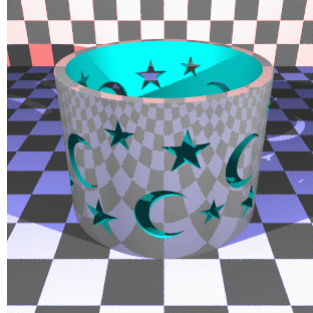
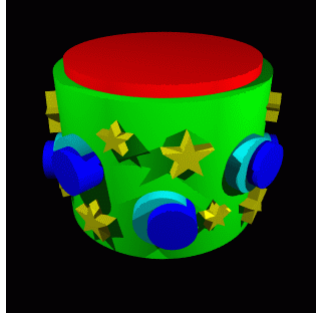


POV-Ray
documentation

Alberto Raposo – PUC-Rio

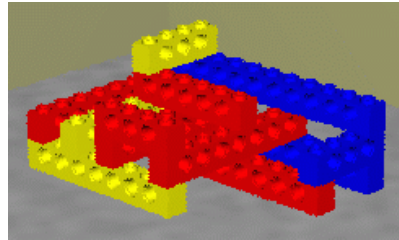


Exemplos CSG



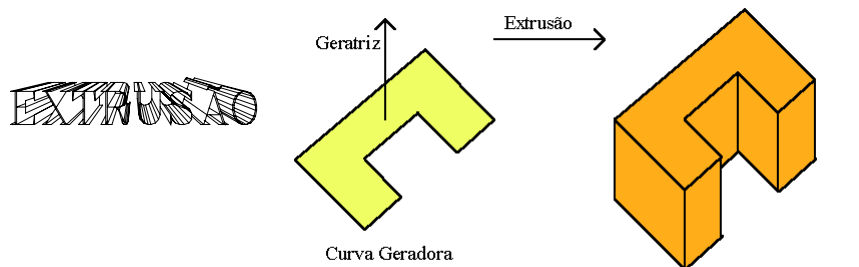
<http://www.cl.cam.ac.uk/Teaching/1998/AGraphics/13a.html>

Alberto Raposo – PUC-Rio



Extrusão

- Acrescenta o eixo z (profundidade) a um polígono

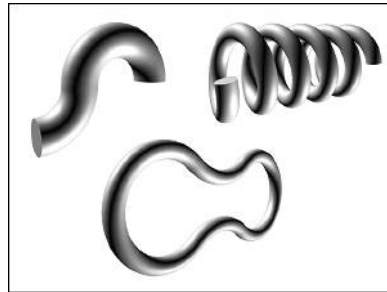
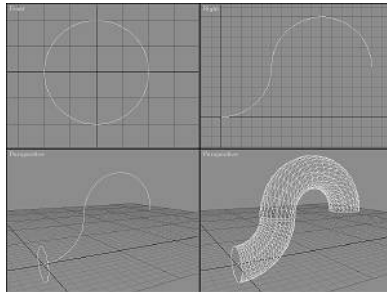


Alberto Raposo – PUC-Rio

Pinho, PUCRS

Sweeping

- Extrusão ao longo de uma curva

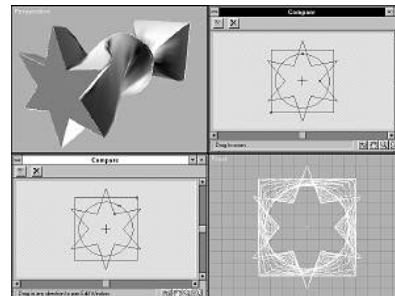
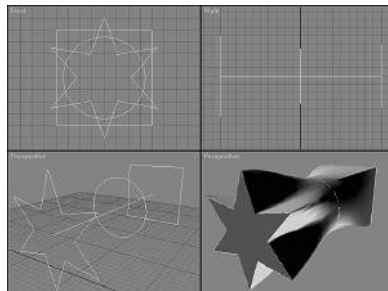


Giambruno, 2003

Alberto Raposo – PUC-Rio

Skinning

- Extrusão ao longo de uma curva (sweeping), mas com cortes variados ao longo do caminho.

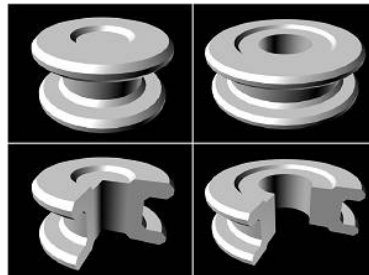
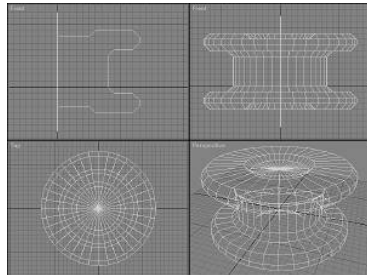


Giambruno, 2003

Alberto Raposo – PUC-Rio

Lathing (sólidos de revolução)

- Sólido é gerado girando superfície em torno de um eixo (ideal para modelos radiais)



Alberto Raposo – PUC-Rio

Giambruno, 2003

VRML: Extrusion

```
Extrusion {  
  beginCap TRUE  
  endCap TRUE  
  ccw TRUE  
  convex TRUE  
  creaseAngle 0  
  crossSection [1 1, 1 -1, -1 -1, -1 0, 1 1]  
  orientation 0 0 1 0  
  scale 1 1  
  solid TRUE  
  spine [0 0 0, 0 1 0]  
}
```

se início / fim da extrusão é aberto

polígono 2D (corte)

curva de extrusão

Alberto Raposo – PUC-Rio

VRML Extrusion - Exemplo

```
#VRML V2.0 utf8
Transform {
```

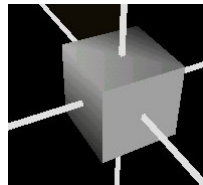
```
  children
```

```
    Shape{ appearance Appearance { material Material {}}
      geometry Extrusion{
```

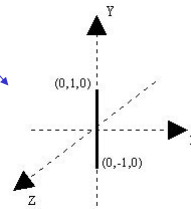
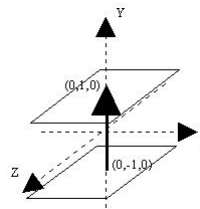
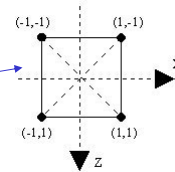
```
        crossSection [-1 -1, -1 1, 1 1, 1 -1, -1 -1]
        spine [0 -1 0, 0 1 0]}
```

```
    }
```

```
  }
```



Alberto Raposo – PUC-Rio



<http://www.lighthouse3d.com/vrml/tutorial/index.shtml?extru>

VRML Extrusion – Exemplo

```
#VRML V2.0 utf8
```

```
Transform {
```

```
  children [
```

```
    Shape{ appearance Appearance { material Material {}}
```

```
      geometry Extrusion{
```

```
        crossSection [ -1 -1, -1 1, 1 1, 1 -1, -1 -1]
```

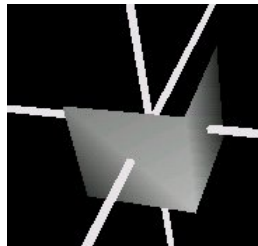
```
        spine [0 -1 0 , 0 1 0 ]
```

```
        beginCap FALSE
```

```
        endCap FALSE}
```

```
    }
```

```
  }
```

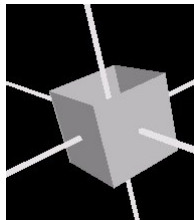


Alberto Raposo – PUC-Rio

<http://www.lighthouse3d.com/vrml/tutorial/index.shtml?extru>

VRML Extrusion - Exemplo

```
#VRML V2.0 utf8
Transform {
  children [
    Shape{ appearance Appearance { material Material {} }
      geometry Extrusion{
        crossSection [ -1 -1, -1 1, 1 1, 1 -1, -1 -1]
        spine [0 -1 0 , 0 1 0 ]
        beginCap FALSE
        endCap FALSE
        solid FALSE}
      }
  ]
}
```



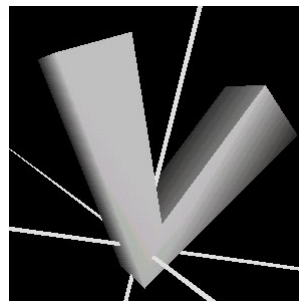
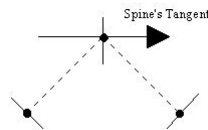
passa a não existir lado interno e externo das faces

Alberto Raposo – PUC-Rio

<http://www.lighthouse3d.com/vrml/tutorial/index.shtml?extru>

VRML Extrusion – Exemplo (sweeping)

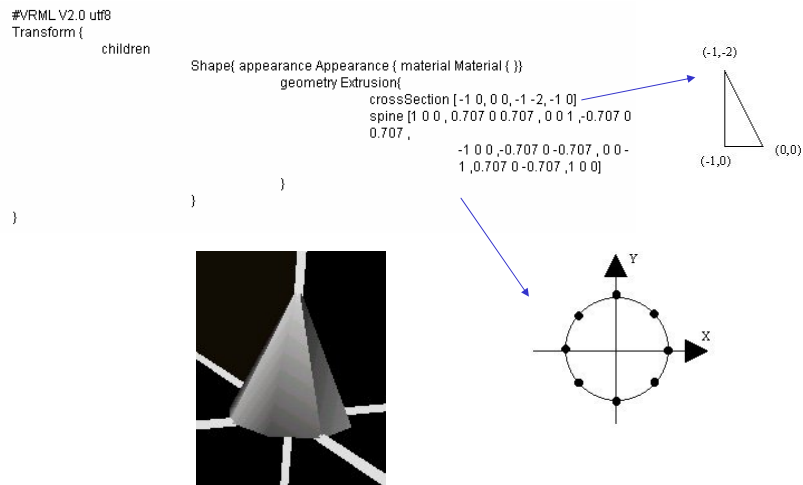
```
#VRML V2.0 utf8
Transform {
  children [
    Shape{ appearance Appearance { material Material {} }
      geometry Extrusion{
        crossSection [-1 -1,-1 1, 1 1, 1 -1,-1 -1]
        spine [3 5 0, 0 0 0,-3 5 0]}
      }
  ]
}
```



Alberto Raposo – PUC-Rio

<http://www.lighthouse3d.com/vrml/tutorial/index.shtml?extru>

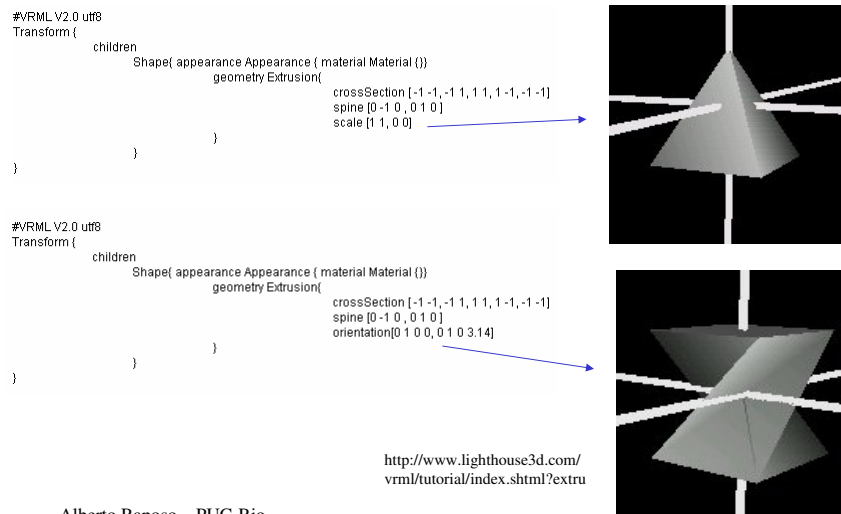
VRML Extrusion – Exemplo (lathing)



Alberto Raposo – PUC-Rio

<http://www.lighthouse3d.com/vrml/tutorial/index.shtml?extru>

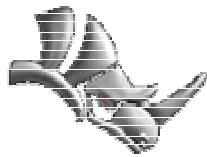
VRML Extrusion – Exemplo



Alberto Raposo – PUC-Rio

<http://www.lighthouse3d.com/vrml/tutorial/index.shtml?extru>

Demo com Software de Modelagem



<http://www.rhino3d.com/>

Alberto Raposo – PUC-Rio

Outras técnicas de modelagem

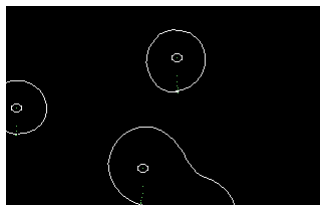
Alberto Raposo – PUC-Rio

Metaballs (Superfícies Implícitas)

- Técnica de modelagem implícita (não paramétrica, como as curvas)
- Modelos gerados a partir de esferas, que podem ser vistas como partículas gerando “campo de atração”, decrescente a partir de seu centro
 - “gosma líquida”

Alberto Raposo – PUC-Rio

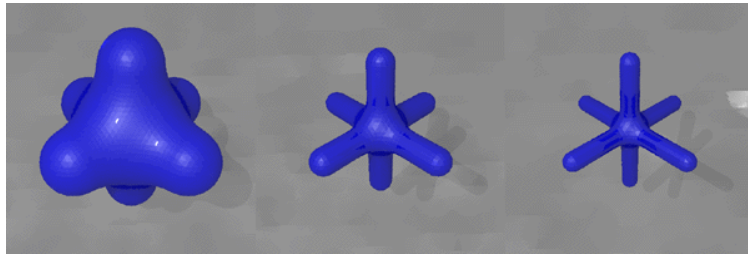
Metaballs



<http://www.niksula.cs.hut.fi/~hkankaan/Homepages/metaballs.html>

Alberto Raposo – PUC-Rio

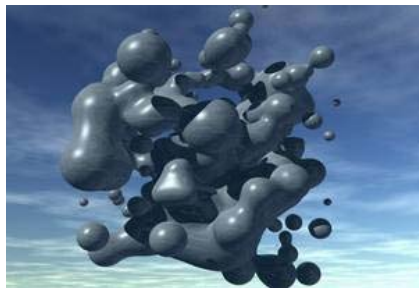
Metaballs



<http://astronomy.swin.edu.au/~pbourke/modelling/implicitsurf/>

Alberto Raposo – PUC-Rio

Metaballs



Alberto Raposo – PUC-Rio

Digital I Designs



Metaballs

- Vantagens:
 - Adequadas para representar metamorfoses e “blendings”

Alberto Raposo – PUC-Rio

Subdivision Surfaces

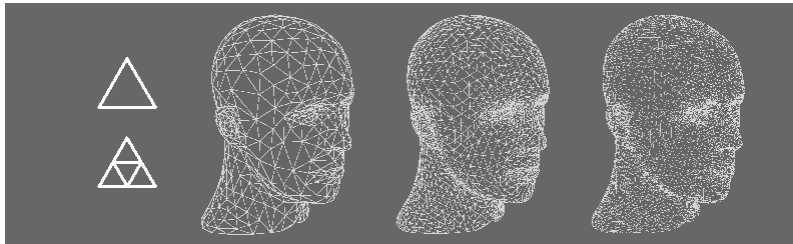
- Nova metodologia de geração de superfícies poligonais “lisas” (smooth), criada pela Pixar para o curta “Geri’s Game”



Alberto Raposo – PUC-Rio

Subdivision Surfaces

- Definição de uma superfície “lisa” como o limite de uma sequência de refinamentos sucessivos

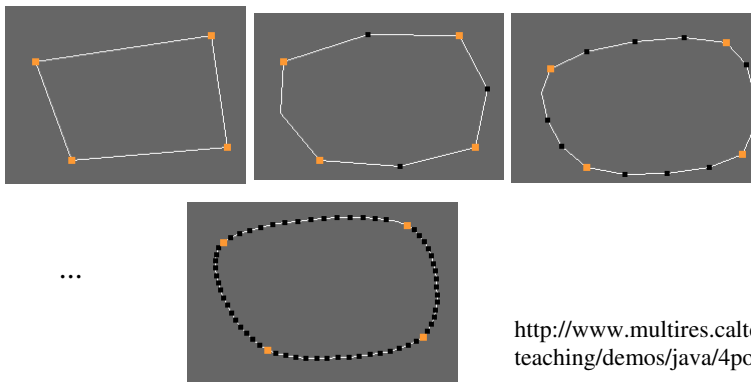


Alberto Raposo – PUC-Rio

<http://www.multires.caltech.edu/teaching/courses/subdivision/intro/index.htm>

Subdivision Surfaces

- Exemplo 2D

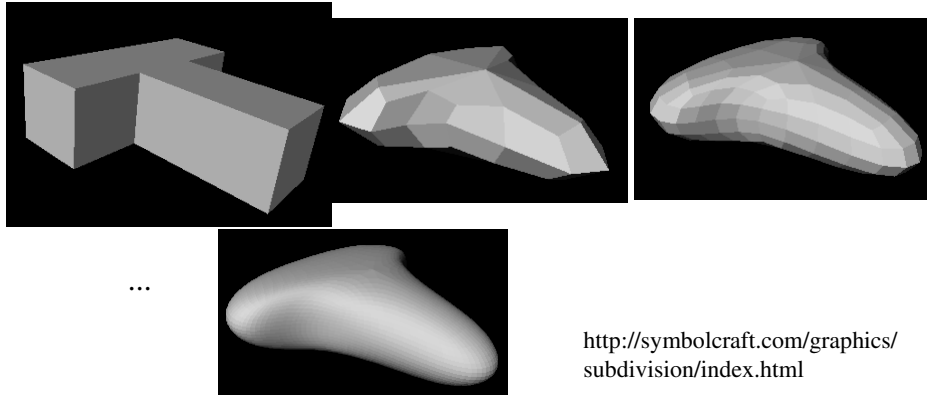


Alberto Raposo – PUC-Rio

<http://www.multires.caltech.edu/teaching/demos/java/4point.htm>

Subdivision Surfaces

- Exemplo 3D

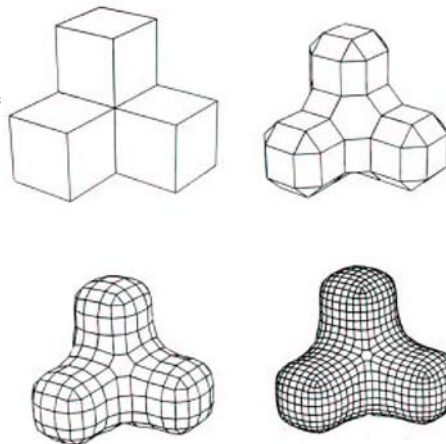
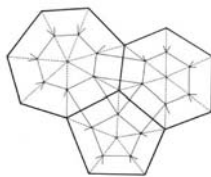


<http://symbolcraft.com/graphics/subdivision/index.html>

Alberto Raposo – PUC-Rio

Doo-Sabin Subdivision

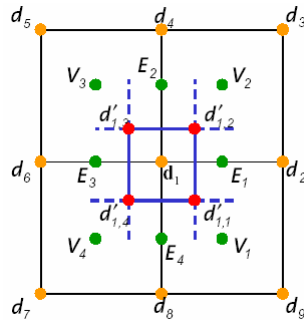
Idéia: introduzir novos vértices em cada face, na metade da distância entre o antigo vértice e o centróide da face.



Alberto Raposo – PUC-Rio

MIT EECS 6.837, Durand and Cutler

Doo-Sabin Subdivision



$$V_2 = \frac{1}{n} \cdot \sum_{j=1}^n d_j$$

$$E_i = \frac{1}{2} (d_i + d_{2i})$$

$$d'_{i,j} = \frac{1}{4} (d_i + E_j + E_{j-1} + V_j)$$

Hatice Çinar
<http://www.cmpe.boun.edu.tr/~akarun/cmpe535projects2004.htm>

Alberto Raposo – PUC-Rio

Vantagens sobre Curvas

- Geração de subdivision surfaces usam algoritmos mais simples que as curvas
- Se encaixam em qualquer topologia, sem problemas de continuidade (muito útil em animação)
- Pode-se representar superfícies com o grau de “smoothness) desejado (escalabilidade, LOD)

Alberto Raposo – PUC-Rio

Low-Poly

- Representações paramétricas, implícitas, subdivision surfaces, etc., buscam modelagem de alta resolução
 - Mais necessidade de processamento
 - Nem sempre adequados para aplicações em tempo real (jogos e realidade virtual, por exemplo).
- Low-Poly: a melhor qualidade possível com número limitado de polígonos



Mole Character ©2001
Barry Collins (model)
Per Abrahamson (mapping)
James Edwards (design)

Alberto Raposo – PUC-Rio

Low-Poly

- Não envolve novas tecnologias de modelagem, mas envolve mais precisão nas tomadas de decisão sobre onde investir em mais detalhes e onde simplificar para obter o melhor resultado
 - Modelagem ruim em alta resolução tem pouco impacto; apenas leva mais tempo para gerar a imagem.
 - Em low-poly, isso é crítico!

Alberto Raposo – PUC-Rio

Low-Poly



<http://www.tutorialized.com/tutorial/Texturing-your-Lara-Croft-model/4859>

Alberto Raposo – PUC-Rio

http://www.muranon.com/axel/character/tutorial_1/

Informações Adicionais

- Modelagem em Geral:
 - D. F. Rogers, J. A. Adams. “Mathematical Elements for Computer Graphics”. 2nd Ed., McGraw-Hill, 1990.
 - Peter Shirley. Fundamentals of Computer Graphics, A K Peters, Ltd., Natick, MA, USA, 2002.
 - Foley, J. D., Van Dam, A., Feiner, S. K., e Huhes, J. F., Philips, L. R., Introduction to Computer Graphics, Addison-Wesley, 1995.
 - <http://www-pal.usc.edu/cs582/index.html>
 - <http://www.inf.pucrs.br/~pinho/CG/Aulas/Modelagem/Modelagem3D.htm>
 - <http://www.ic.uff.br/~aconci/sweeping.html>
 - <http://www.inf.unisinos.br/~osorio/CG-Doc/CG-Web/cg.html>

Alberto Raposo – PUC-Rio

Informações Adicionais

- Quadtrees e Octrees
 - <http://www.tecgraf.puc-rio.br/~hermann/gc/>
 - http://www.flipcode.com/tutorials/tut_octrees.shtml
- LOD
 - D. Luebke, M Reddy et al. “Level of Detail for 3D Graphics”. Morgan Kaufman, 2003.
 - T. Möller, E. Haines. “Real-Time Rendering”. A K Peters Ltd., 1999.

Alberto Raposo – PUC-Rio

Informações Adicionais

- Metaballs:
 - G. Graves. “The Magic of Metaballs”. Computer Graphics World, Maio 1993.
 - <http://astronomy.swin.edu.au/~pbourke/modelling/impliciturf/>
- Subdivision surfaces:
 - <http://www.eas.asu.edu/~cse470/resources/subdivision/>
 - <http://mrl.nyu.edu/publications/subdiv-course2000/>
- Low-Poly:
 - M. Giambruno. “3D Graphics & Animation”. New Riders, 2002
- The Annotated VRML 97 Reference:
<http://accad.osu.edu/~pgerstma/class/vnv/resources/info/AnnotatedVrmlRef/Book.html>

Alberto Raposo – PUC-Rio